It's hard for me not to write way too much on this subject, but AI / balancing / difficulty was the biggest challenge in making the game. The important thing to remember about Into the Breach's design philosophy is that the game is more of a puzzle game than a strategy game. Things don't just get "hard" they get "unsolvable" if everything isn't working correctly. And unsolvable is a really frustrating state to find yourself in as a player. So every decision made reflects that in some way, even if I'm not stating it explicitly.

I also tend to prefer extremely simple (bordering on "dumb") solutions for nearly everything if possible. It's easier to implement, maintain, edit, extend, etc. There are definitely more complex options for the AI in Into the Breach, even options that may have made a "better" game. But as the sole programmer on the team, "good enough" is generally what I'm aiming for.

Basic AI logic works like this, and early versions of the game for years *only* used this:

1. Compile everything that the unit can do

The game takes every tile that the unit can move to, and every "target tile" that its weapon can shoot at from that position. So in the shot above, there's roughly 30 tiles that the unit could move to and there will be 4 target tiles at each location (up, down, left, right from the position).

2. Score each of those "moves"

Above, you can see the “target score” for the hornet from that position. It gets points per enemy unit, building, or friendly unit that the attack will hit. While it can technically vary per enemy, so different enemies have different behaviors, 95% of them just use 5 pts, 5 pts, and -2 pts respectively for those types of targets.

The lower scored targets are dropped, and the visual is showing you the “best” target score at each location. So hitting 2 buildings is earning it 10 points (5x2) for the attack.

3. Select a move

Simply order all the possible actions by their score, and randomly pick from between “the best”.

==========

We then expanded the AI slightly as needed for its scoring system, and these were the additional constraints we used:

Non-Optimal Behavior:

The AI can choose from the “second best” options as well. This is especially useful for difficulty settings (having the ‘optimal’ choice picked more often on harder difficulties).

Don’t repeat behavior:

It’s not fun to have a Scorpion web + queue up an attack on your mech, punch him backwards, then just have him come right back at you. So I negatively weight repeated positions / targets.

Location Scores:

Score each tile that the enemy ended up separately from its “target score”, creating a “location score” as well.

The “location score” and the “target score” are not just added together. Instead it follows this criteria:

- If the location score is negative for any reason, the game will immediately drop that choice. Mostly this reflects “dangerous” tiles/movements.
- If the target scores are all 0 (it cannot find something to shoot), fall back on ranking via the location score.

The actual location score is calculated like this:

Melee Units: These prefer to be “close” to an enemy, so there is weighting to make them move closer even in situations. This is specifically for when they can’t hit anything.

Dangerous: Fires, pits, enemy attacks, etc. are all “dangerous” and the enemy attempts to avoid them and are negatively weighted.

Edges: The game is really unpleasant when an enemy ends up on the edges, you cannot push them or manipulate them nearly as effectively. So we heavily negatively weight the edges, though enemies *can* end up there to avoid fire and other dangers.

==========

Players often assume the AI is doing many things more than it actually does. As these come up often, I like to mention explicitly that we never do the following:

1. Enemies don’t care what other enemies are doing. They don’t pre-plan to make sure they spread out attacks, etc. However, if an enemy earlier in the turn queue is attacking a tile, that tile will be considered “dangerous” for the following enemies during their calculations.

2. Enemies don't care if it's a building or a mech they are attacking, and do not try to distribute their attacks between them.

3. Enemies do not care what weapons, mods, etc. the player is using, or if the situation they are creating is "solvable"

4. Enemies *never* plan future moves, as you might for a chess AI or other similar game AI's.

==========

And I think it's also important to note that the actual AI is only the start of balancing difficulty / enemy behavior. We also rely on other things to control the experience for the player:

Map Design: All of the maps are handmade, with a few random elements (like where forests end up). The map layout is *very* important for creating "solvable" problems. If there are closed in corners and other terrain features that are difficult to manipulate the enemy around, it's a risk. It's easier to design the maps *not* to have those than to teach an AI to avoid them.

Spawn Management: How many units spawn and what types of units spawn was the most important part of Into the Breach balancing. The game is really on a knife edge where it can turn from a "solvable puzzle" to an impossible nightmare. And that difference is often decided by just adding 1 extra unit, resulting in 1 extra problem, which your limited 3 actions possibly can't deal with. I'm not going to dig into the logic behind it, but it was a long process of playtesting and iteration to get that balance right. And the game adapts to the player depending on how a mission is going (kill a lot of units and spawns will increase to replace them) -- the goal for us was to always keep every turn as interesting as possible, neither too easy nor too difficult.